PREDICTING PROTEIN-PROTEIN INTERACTION VIA CONVOLUTIONAL ADAPTIVE DOT PRODUCT

A Dissertation
Presented to
The Academic Faculty

By

Diptodip Deb

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in the
School of Computer Science

Georgia Institute of Technology

May 2018

PREDICTING PROTEIN-PROTEIN INTERACTION VIA CONVOLUTIONAL ADAPTIVE DOT PRODUCT

Approved by:

Dr. James Hays, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Annalise Paaby
School of Biological Sciences
Georgia Institute of Technology

Date Approved: May 04, 2018

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

We propose a deep-learning method for predicting protein-protein interaction (PPI) via an adaptive dot product in combination with dilated convolutions. Protein-protein interaction is a crucial biological process. Being able to predict when two proteins will interact with each other can be crucial information for developing drug targets. Adaptive convolutions have proven effective at determining some relationship between two inputs in the context of video interpolation, and dilated convolutions in WaveNet have proven effective at understanding sequence information in the context of audio generation. Our proposed method combines the dilations in one dimension to interpret sequence information from a WaveNet approach (though without the conditional nature) and uses it to produce an adaptive dot product kernel while simultaneously learning an embedding of the input via an autoencoder. The dot product of the generated kernel and embedding can then be used to predict PPI. We show that this method is able to extract useful features and obtain performance very near the state-of-the-art that uses hand-engineered features that take advantage of human knowledge of proteins. We anticipate that this method will more generally be an efficient approach to any problem requiring sequence comparison.

CHAPTER 1
INTRODUCTION

1.1 Background

Protein-protein interactions (PPIs) are crucial to a variety of biological processes, e.g. metabolism or replication of DNA [1, 2, 3, 4]. Misregulation of protein-protein interactions can be the attack mode of various diseases. Designing treatments for these disease systems, which can include cervical cancer, bacterial infection, leukemia, and neurodegenerative diseases, has been the focus of various biological studies [1]. Understanding the network of interactions between proteins can not only help to understand or identify the biological processes occuring in a cell, they can also assist in the choice of drug targets or drug design by making use of interacting proteins or avoiding side effects of interacting proteins [4, 1]. Indeed, various approaches of tackling such disease systems have made use of protein-protein interaction inhibitors [1].

There have been a number of experimental procedures to determine PPI, many of which are high-throughput [4, 2]. These approaches include tandem affinity purification, which creates a fused protein that is allowed to interact with other proteins and is then removed, allowing for the analysis of the remaining interacting proteins via gel electrophoresis [2, 5]. Other approaches include affinity chromatography [2, 6], Co-Immuno-precipitation (Co-IP)[2, 7], X-ray crystallography[2, 8], nuclear magnetic resonance imaging (MRI scans) [2, 9], and the Yeast Two-Hybrid system [2, 10]. Another modern method combines quantitative proteomics with spatial references and the use of APEX, a peroxidase that has allowed capturing of proteomes with high temporal resolution, to discover protein interaction networks with high temporal and spatial resolution.

All of these experimental methods attempt to separate out a batch of proteins that have

interacted with a protein by either chemical or imaging means. Due to this experimentation, the availability of PPI data has significantly increased [4, 2, 11, 12, 13, 14, 15]. However, almost any experimental method is costly and time-consuming, especially when compared to computational methods. As a result, the experimental data that exists can only illuminate fragments of the full PPI networks [4]. Furthermore, these high-throughput methods for PPI are prone to failure (false positives and negatives) [4, 2]. The motivation for accurate computational methods to predict PPI are therefore clear. A computational prediction could be cheap, quick, and could provide predictions for the full network of proteins.

Initial applications of computational approaches to the PPI problem were based on human knowledge of proteins [2]. Various studies were conducted that either attempted to gather properties of interacting proteins or use such known properties to build deterministic systems that found likely candidates for specific types of interacting proteins [2]. One such approach compared the observed properties of fused genes, that is understanding the properties of proteins resulting from a fusion of the genes that encoded them into a single chain, because the resulting fused proteins would clearly interact [2, 16]. Other approaches involved showing that proteins produced by conserved gene neighborhoods, clusters of genes that overlapped across various genomes, were likely to interact in certain bacteria and archaea [2, 17]. Other approaches characterized known protein interaction networks and found that they exhibited scale-free and small-world topologies [2, 18]. These knowledge based techniques were still relatively limited compared to the modern methods of predicting protein-protein interaction, and relied on information about the functional domains and pathways of the relevant proteins as well as their encoding genes [2].

Modern methods for PPI prediction are concerned with classifying interaction using machine learning systems that take in features based on the amino acid sequence [3, 4]. These methods all typically involve the extraction of relevant statistical and physiochemical features from the amino acid sequence by hand using a variety of methods involving either local descriptors, multi-scale local descriptors, correlations, mutual information between

the sequences, and others [3, 4]. These hand-engineered feature vectors are then used as input to various standard machine learning algorithms, such as random forests or multilayer perceptrons [3, 4, 2]. However, this challenge of extracting features is difficult: protein interactions are often due to discontinuous amino acid segments in the sequence. This means taking the effect of amino acids from across the entire acid sequence into account is important [3, 4]. The features used must encode global information from the rather noisy input of amino acid sequences [2], which demonstrates the difficulty of using machine learning within the PPI problem.

1.2 Motivation

Given the difficulty of experimentally determining protein interaction networks, computational approaches are clearly desirable [2, 4, 3]. Machine learning approaches have been shown to have high accuracy on various datasets and can do so using information from the amino acid sequence alone [2, 3]. However, all of these methods rely on hand-engineered feature vectors that are constructed with domain knowledge of how proteins are constructed physically and chemically as well as how they interact. While this is certainly an area in which human domain knowledge might be crucial for high accuracy, it is clear that continuing to design better features is a difficult task and will not scale as well as using better machine learning algorithms [2]. Deep learning has been shown to have a great amount of success on various difficult tasks [19, 20, 21, 22, 23, 24, 25]. Indeed, DeepPPI shows that a deep learning network can achieve state of the art performance on the PPI task for various standard datasets [2].

The DeepPPI method is to simply use a two-input multilayer perceptron that takes feature vectors from both amino acid sequences as input [2]. This shows that deep learning is able to effectively combine the protein features for classification. Their approach achieves state of the art performance, yet it fails to take advantage of the representation learning capability of deep learning because it relies on a hand-engineered feature vector[19, 26].

While domain knowledge may be necessary in the difficult task of PPI prediction due to the noisy input of an amino acid sequence, minimizing the need for designing hand-engineered features is still diesirable because with enough data and an effective representation learning method, hand-engineered feature methods can be beaten out [19, 26]. Traditionally when sequence data is involved, deep learning approaches turn to recurrent neural networks [27, 28]. Instead, we propose to use convolutions. Convolution based approaches have two advantages here: (1) they are much faster than recurrent methods and (2) as noted, interaction patterns are often not in continuous sections of the amino acid sequence [27, 2]. Because convolution approaches are spatially invariant, convolutions for might allow for more effective capturing of interaction patterns with a much lower memory footprint compared to a recurrent method. We look to two convolutional networks for inspiration: the 1D PixelCNN of WaveNet [24] and adaptive convolutions for video interpolation [29]. The success of the 1D PixelCNN in learning a conditional probability distribution from sequential input and the success of adaptive convolutions in combining information from two frames of video to interpolate the intermediate frame should allow for the PixelCNN inspired dilations of our method to effectively be able to generate the adaptive kernel used to take the final dot product across the sequence embedding that results in the classification, as described in Section 1 of Chapter 3.

CHAPTER 2
LITERATURE REVIEW

2.1 Previous machine learning methods for PPI

There are various machine learning methods that have achieved good results on the PPI problem. These include the multi-scale local descriptor (MLD) and multivariate mutual information (MMI) [3, 4]. Both of these approaches produce a feature vector which is given as input to a random forest classifier. The MLD approach involves extracting physicochemical features from multi-scale continuous segments of amino acids, which allows this approach to capture multiple overlapping patterns of interaction in the amino acid sequence [4]. The MMI approach involves clustering the amino acids into 7 functional groups and transforms the sequences of amino acids into sequences of these encodings. Then, the mutual information is calculated between these two sequences. Combining this mutual information with an autocorrelation approach, a feature vector is produced which is passed to a random forest classifier [3].

Other machine learning approaches to the PPI problem include the use of local phase quantization descriptors and rotation forests [30]. This approach also uses phsyicochemical properties of the proteins in order to construct features. Because random forests would normally be sensitive to rotation of the features constructed, this method uses rotation forests instead [30]. We note that all of these state of the art machine learning approaches simply use some form of ensemble learning via forests and vary mostly in their choice of feature engineering.

Deep learning methods have also been applied to the PPI prediction problem. However, these networks do not take advantage of the representation learning power of deep learning; the sole deep learning methods employed for this problem have been simple multilayer

perceptrons [31, 2]. Like other modern methods for PPI prediction, these networks are concerned with classifying interaction based solely on the amino acid sequence [3, 4], yet the networks in question also operate via the extraction of hand-engineered feature vectors from the amino acid sequence using a variety of methods involving either local descriptors, multi-scale local descriptors, correlations, mutual information between the sequences, and others [3, 4, 2].

Therefore, the deep learning methods face the same problem as the forest methods: the challenge of extracting features is difficult. Protein interactions are often in discontinuous amino acid segments in the sequence due to the complex 3D folding of the tertiary structure of a protein. This means taking the effect of neighboring amino acids into account is difficult [3, 4]. Methods that continue to rely on hand-engineered features will be limited by those features while the availability of PPI data will continue to grow. Therefore, it is prudent to learn to extract the appropriate features while manipulating the amino acid sequence as little as possible, i.e. minimizing the need for feature engineering.

2.2 Convolutions for feature extraction

The precedent of using neural networks to learn representations of biological sequences is not new. Based on the idea of Word2Vec [32], ProtVec trains an autoencoder to predict an amino acid based on the surrounding amino acids in order to learn an embedding of amino acid sequences [33]. In our case, however, we are interested in learning not just the representation of a single amino acid, but learning a good representation of two amino acids for comparison.

Convolutional neural networks have been shown to effectively extract features in the context of images [19, 20, 21]. More specifically, Siamese CNNs have been shown to be able to compare two input images and report their similarity [34]. These networks function by using two parallel networks that are combined and trained together for the image verification task, which is used for oneshot learning. The structure of this network is ac-

tually similar to the two-input multilayer perceptron used in DeepPPI [2]. For interpreting sequence information, convolutions have also proved useful as shown by WaveNet, [24]. WaveNet uses convolutions that are dilated in a single dimension in order to extract features from sequence input. The convolutions maintain the order of sequences by masking the convolution kernels appropriately, though this is not as relevant for amino acid sequences. Dilations were defined in [35] and shown to efficiently increase the receptive field of a convolutional neural network.

The standard convolution operation as used in deep learning is defined by [35]. We first define a real valued function $F : \mathbb{Z}^2 \to \mathbb{R}$, an input $\Omega_r = [-r, r]^2 \in \mathbb{Z}^2$, and a filter function $k : \Omega_r \to \mathbb{R}$. A convolution operation is therefore given by

$$(F * k)(\mathbf{p}) = \sum_{\mathbf{s}+\mathbf{t}=\mathbf{p}} F(\mathbf{s})k(\mathbf{t}). \tag{2.1}$$

Dilating a convolution is a generalization of a convolution. This involves introducing a dilation parameter which can allow the convolution operation to stretch the kernel size while skipping some inputs. Therefore, fewer parameters are required for a kernel to operate over a larger input size. The definition as given in [35] is as follows:

$$(F *_l k)(\mathbf{p}) = \sum_{\mathbf{s}+l\mathbf{t}=\mathbf{p}} F(\mathbf{s})k(\mathbf{t}) \tag{2.2}$$

where l is the index of the current layer of the convolution.

Thus, we see that deep learning is able to handle comparison of two inputs as well as extract useful sequence information.

The work of [29] uses adaptive convolutions in order to generate the in-between frame of video given the before and after frames of video. This is another convolutional neural network that compares two inputs, but does so by generating a convolution kernel based on the data. For each pixel position, the network uses convolutions to extract features from the two frames (represented as two channels) and output to a kernel. Then, the kernel is

convolved at that pixel position for the two input frames. The result is summed to obtain the final result. The results suggest that using adaptive convolutions could allow for better extraction of features when the task of interest involves the relationship of two inputs [29].

Overall, machine learning methods have proven effective at the PPI problem, but rely too much on hand-engineered features [2, 4, 30]. Various results in deep learning have shown that convolutions, applied appropriately, can be useful feature extractors [29, 24, 35, 34].

CHAPTER 3

METHOD

3.1 AdaDot Architecture

Taking inspiration from [29], we want to generate a kernel from the input. However, unlike the task of video interpolation in which local regions between frames will not change much, the PPI task requires use of more global information. Preliminary experiments showed that taking a windowed approach to generating kernels (such that the output of each window essentially cast a "vote" on the final output) resulted in poor classification performance. From these experiments, we determined that it is necessary to construct a good embedding of the two input proteins prior to weighting the result with the generated kernel.

Based on the work of [24] and [35], we dilate sequential convolution layers at increasing dilation rates (specified in Table 3.1) which should provide a large receptive field. This is ideal for extracting global features from proteins, which is necessary because protein interaction between two proteins can occur between highly distant regions of the amino acid sequence.

The overview of the full architecture is shown in Figure 3.1, and the specific parameters of the layers are shown in Table 3.1. All layers have a ReLU activation applied to them. The goal of AdaDot to simultaneously learn a good representation of the input proteins and to generate the appropriate kernel. In order to accomplish this, AdaDot learns two tasks simultaneously: classification of PPI and reconstruction of the input sequences.

The input I is a two channel $n \times s$ tensor where s is the length of the amino acid sequence and n is the number of encodings of the sequence. Each channel represents one of the two amino acids. The acids can be reencoded based on whether we want to use the raw amino acid sequence as input (using 21 unique index values from 1 to 21 to represent

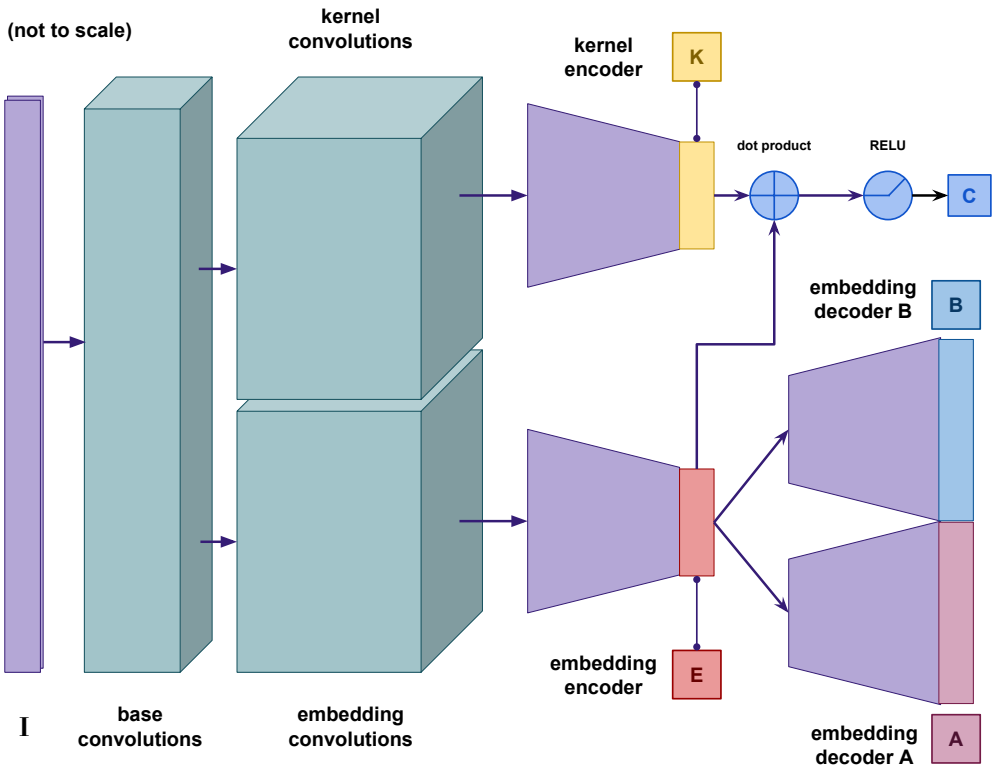

Figure 3.1: Architecture of the proposed AdaDot network. More details are available in Table 3.1.

Group	Layer Order	Layer Type (*H. pylori*)	Layer Type (*S. cerevisiae*)
base convolutions	1	$Conv(32, (5, 7), 1)$	$Conv(32, (7, 9), 1)$
base convolutions	2	$Conv(32, (5, 7), 1)$	$Conv(32, (7, 9), 1)$
base convolutions	3	$Conv(32, (5, 7), 2)$	$Conv(32, (7, 9), 2)$
embedding convolutions	1	$Conv(32, (5, 7), 4)$	$Conv(32, (7, 9), 4)$
embedding convolutions	2	$Conv(32, (5, 7), 8)$	$Conv(32, (7, 9), 8)$
embedding convolutions	3	$Conv(32, (5, 7), 16)$	$Conv(32, (7, 9), 16)$
embedding convolutions	4	$Conv(32, (5, 7), 32)$	$Conv(32, (7, 9), 32)$
embedding convolutions	5	$Conv(32, (5, 7), 1)$	$Conv(32, (7, 9), 1)$
embedding convolutions	6	$Conv(1, (5, 7), 1)$	$Conv(1, (7, 9), 1)$
kernel convolutions	1	$Conv(32, (5, 7), 2)$	$Conv(32, (7, 9), 2)$
kernel convolutions	2	$Conv(32, (5, 7), 2)$	$Conv(32, (7, 9), 2)$
kernel convolutions	3	$Conv(32, (5, 7), 4)$	$Conv(32, (7, 9), 4)$
kernel convolutions	4	$Conv(32, (5, 7), 8)$	$Conv(32, (7, 9), 8)$
kernel convolutions	5	$Conv(32, (5, 7), 1)$	$Conv(32, (7, 9), 1)$
kernel convolutions	6	$Conv(1, (5, 7), 1)$	$Conv(1, (7, 9), 1)$
embedding encoder	1	$Dense(2048)$	$Dense(2048)$
embedding encoder	2	$Dense(1024)$	$Dense(1024)$
embedding encoder	3	$Dense(512)$	$Dense(512)$
embedding encoder	4	$Dense(256)$	$Dense(256)$
embedding decoder A	1	$Dense(512)$	$Dense(512)$
embedding decoder A	2	$Dense(1024)$	$Dense(1024)$
embedding decoder A	3	$Dense(2048)$	$Dense(2048)$
embedding decoder A	4	$Dense(s)$	$Dense(s)$
embedding decoder B	1	$Dense(512)$	$Dense(512)$
embedding decoder B	2	$Dense(1024)$	$Dense(1024)$
embedding decoder B	3	$Dense(2048)$	$Dense(2048)$
embedding decoder B	4	$Dense(s)$	$Dense(s)$
kernel encoder	1	$Dense(1024)$	$Dense(1024)$
kernel encoder	2	$Dense(512)$	$Dense(512)$
kernel encoder	3	$Dense(256)$	$Dense(256)$

Table 3.1: Detailed parameters of the proposed AdaDot network. $Conv(f, (k, l), d)$ represents a convolutional layer producing f filters using kernels of shape $k \times l$ and a dilation rate of $1, d$ where $d = 1$ represents no dilation. $Dense(k)$ represents a fully connected perceptron layer consisting of k perceptrons. We let s represent the length of the input sequence.

the amino acids) or if we want to group the amino acids together according to their various biochemical properties. In the first case, $n = 1$ because we are only using the raw amino acid sequence. However, if we group the amino acid sequences together based on various properties, we can come up with various different representations of the same amino acid. This results in an input with multiple "rows." For example, if we grouped the sequence of amino acids "KEDGTQVIMAS" by hydrophobicity according to Table A.1, we would encode it as "11122133322." In all cases, we divide the resulting vector by the size of the alphabet. Sequences are also padded with zeros such that they are all as long as the longest sequence. This can be helpful in order to give some more biochemical information to the network. If the input has more than one row, during reconstruction we simply reconstruct the first row as the amino acid sequence. These various groupings of amino acids are included in Table A.1. We perform experiments described in Section 3.2 to show that better encodings of the amino acid sequence are benefitial to the performance of AdaDot on the PPI task.

These are then passed to a series of base convolutions which extract some common base features. From there, two parallel columns of convolutions operate on those base features. One column is flattened and attached to an autoencoder to produce the embedding E, and the other column is flattened and attached to a series of perceptron layers to produce the weighting kernel K. The size of the kernel matches the size of the embedding. For the final output C, we take $K \cdot E$ and then apply a ReLU activation. The outputs of the autoencoder are A and B, which are the reconstructions of the two amino acid sequences.

The final output is the result of the dot product of the generated kernel K with the generated embedding E as shown in Figure 3.1. This dot product is then activated with ReLU. Classification is defined such that a value greater than 0.5 signifies an interacting pair and a value below 0.5 signifies a non-interacting pair.

During training, we use the Adam optimizer with a learning rate of 10^{-6}. The loss function used is mean absolute error for all three outputs as shown in Figure 3.1.

3.2 Experiments

We experiment on two standard PPI datasets as defined in a variety of PPI papers [2, 4, 30]. These include the *H. pylori* and *S. cerevisiae* PPI datasets. We select these datasets because they have been constructed to have a balanced number of interacting and non-interacting examples.

We first experiment on the smaller *H. pylori* dataset using a variety of architectures to provide some experimental justification of the AdaDot architecture. That is, we show that AdaDot can outperform some standard benchmark CNN architectures. These standard CNN architectures use a series of convolutions that match the last two base convolution layers followed by the embedding convolution layers. We also test on a version of this architecture that has no dilations. Both of these standard CNN architectures end by flattening the results of the convolutions and then connecting to a series of perceptron layers of size 32, 16, 4, and 1 in respective order. All of these layers are activated using ReLU.

We perform these experiments using both the raw amino acid sequence and 8 of the groupings defined in Table A.1. The specific 8 groupings we use are shown in Table A.1. We also use all 24 of the groupings shown in Table A.1, though we only perform this test for the AdaDot architecture. When performing these experiments, we split the data randomly such that 75% is used for training and 25% for testing. Within the training set, we also randomly hold 33% for validation.

We then train and test on both the *H. pylori* and *S. cerevisiae* datasets. The standard testing method is to use k-fold cross validation and report the average error on each of 5 random splits of the data [2, 4, 30]. We follow this standard when reporting accuracies. We also use the 8 groupings to represent the amino acids as mentioned above when reporting these scores.

3.2.1 H. pylori dataset

We use the *H. pylori* dataset as described by [2] and [36]. This dataset consists of 2916 protein pairs and is evenly split between interacting and non-interacting examples. From the description in [36], the dataset is constructed by choosing the explicitly interacting protein pairs as interacting proteins and the remaining pairs as non-interacting proteins as described in the protein interaction map provided by [37].

3.2.2 S. cerevisiae dataset

We use the *S. cerevisiae* "core" dataset as described by [2] and [4]. This dataset originates from the Database of Interacting Proteins (DIP) [38]. As described in [4, 39], the dataset is evenly split between interacting and non-interacting examples. The dataset is constructed by removing the protein sequences that had fewer than 50 residues and protein pairs that had greater than 40% sequence identity (proportion of sequence that matches its pair) [4]. This leaves 5594 interacting protein pairs. To construct 5594 non-interacting protein pairs for this dataset, first known non-interacting protein pairs are selected. From these, only 5594 protein pairs whose subcellular localization is different were selected.

CHAPTER 4

RESULTS

4.1 Experiments on *H. pylori*

We see in Figure 4.1 that AdaDot outperforms standard CNN approaches on the *H. pylori* dataset. The bar graph in Figure 4.1 includes the expected accuracy (of 0.5) with random guessing to show that CNN approaches are indeed able to learn useful features on the PPI task. However, we see that the standard CNN approach only marginally improves on random guessing. Surprisingly, dilated CNNs using the raw amino acid sequence are able to greatly improve over random guessing. However, we see that the combination of AdaDot with a reencoding using 8 feature groupings is able to beat the performance of dilations alone. We do observe that using all 24 features increases the accuracy, however this requires an increased convolution kernel size. It appears that the standard dilated CNN is not able to make effective use of the 8 groupings and loses accuracy.

We see in Figures 4.2 to 4.4 that AdaDot learns different embeddings, on average, for interacting protein pairs as compared to non-interacting protein pairs. This indicates that the autoencoder does manage to force the network to learn some difference in embedding between interacting and non-interacting pairs. It appears that there are a few key elements in the embeddings that do not vary depending on whether the two proteins interact or not.

4.2 Comparison with other methods

We see that in general, deep learning methods have not been able to beat the ensemble method employed by [4]. However, we see that both deep learning methods are close to the state-of-the-art, and notably that AdaDot manages to achieve this performance with minimal feature engineering.

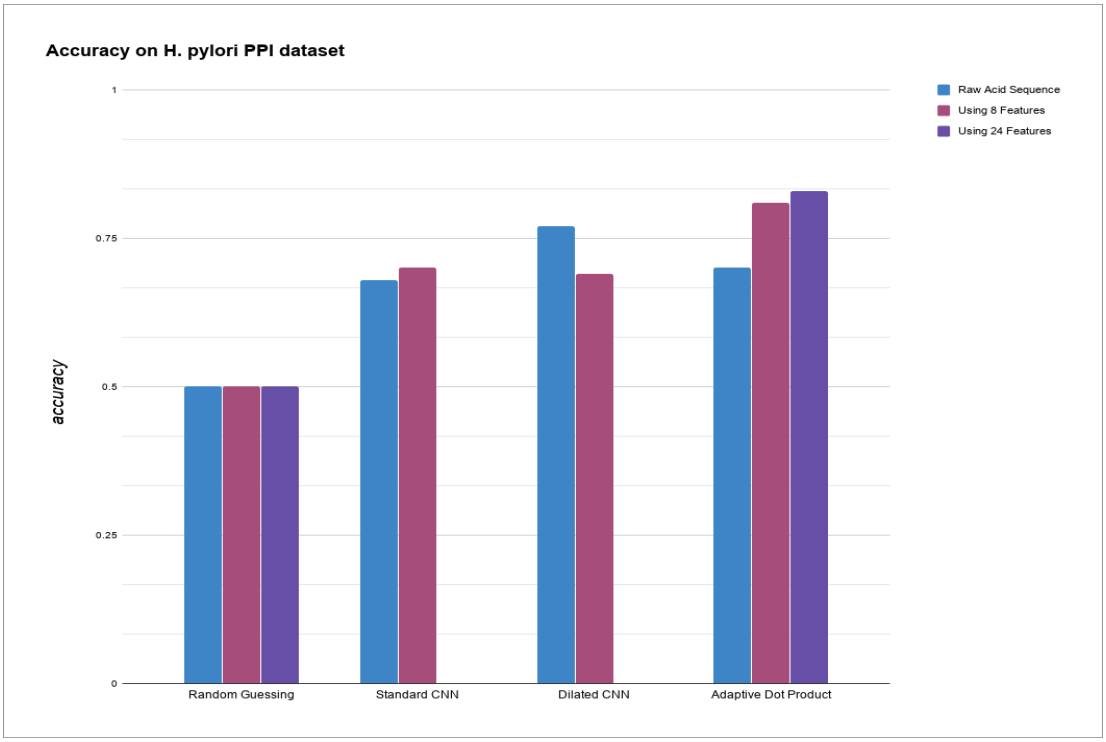

Figure 4.1: Results using various methods on *H. pylori* dataset to justify AdaDot architecture

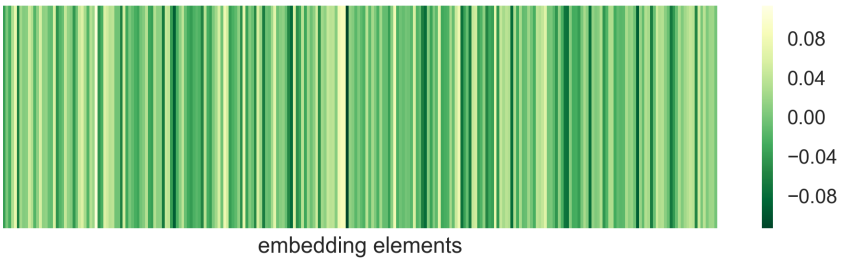

Figure 4.2: Visualization of average interacting protein pair embeddings from the *H. pylori* dataset where the embedding elements from 1 to 256 increase along the bottom axis and color represents the value of the activation

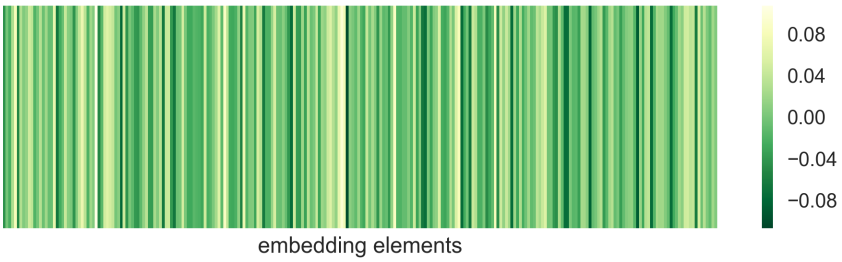

Figure 4.3: Visualization of average non-interacting protein pair embeddings from the *H. pylori* dataset where the embedding elements from 1 to 256 increase along the bottom axis and color represents the value of the activation

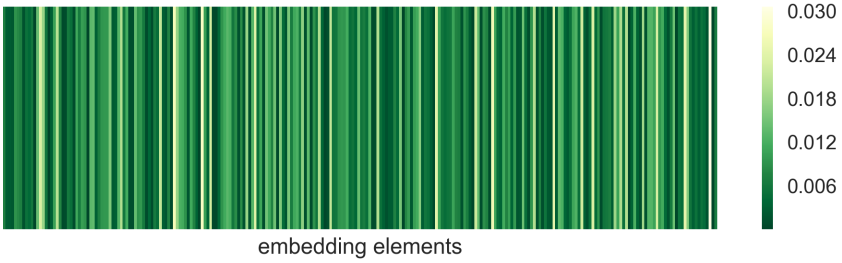

Figure 4.4: Visualization of average absolute difference of embeddings of interacting and non-interacting protein pairs from the *H. pylori* dataset where the embedding elements from 1 to 256 increase along the bottom axis and color represents the value of the activation

Method	**Accuracy**
AdaDot (ours)	0.8133
DeepPPI [2]	0.8623
[4]	**0.8830**
[40]	0.8491
[41]	0.8750
[42]	0.8674
[43]	0.8420
[44]	0.7580
[45]	0.8400
[36]	0.8340
[46]	0.8660

Table 4.1: Performance of AdaDot compared to various state-of-the-art methods for PPI on *H. pylori*

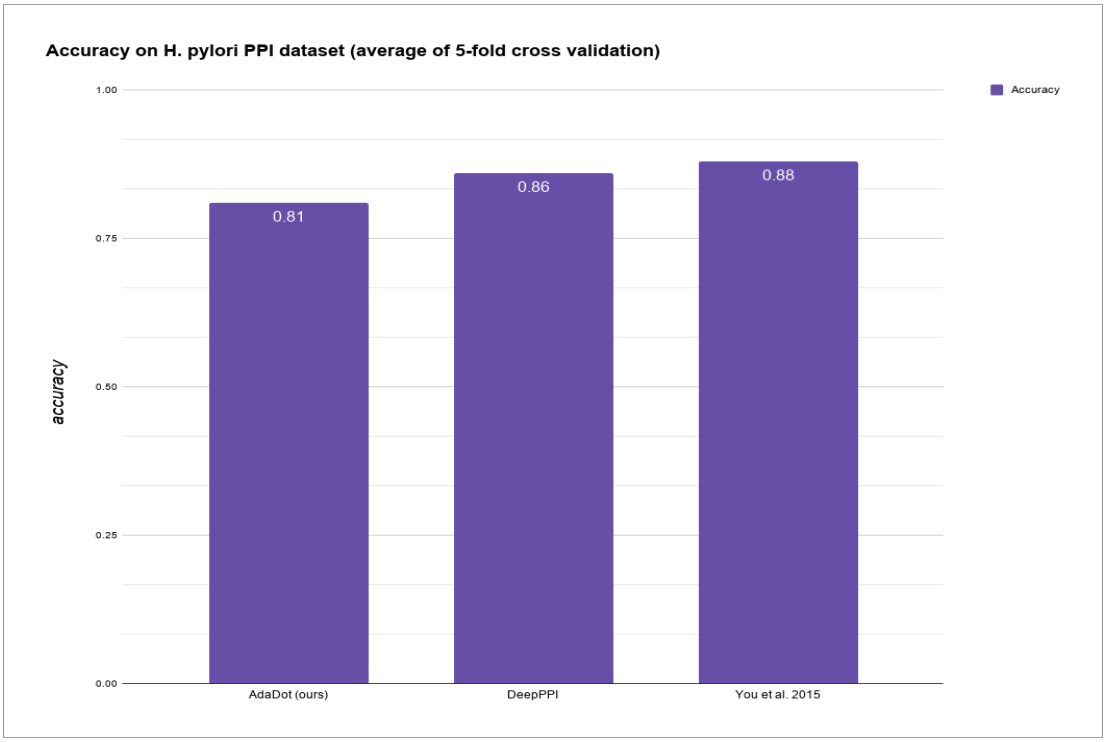

Figure 4.5: Comparison of performance of AdaDot to DeepPPI and the top non-deep method on *H. pylori*

Method	Accuracy
AdaDot (ours)	0.8441
DeepPPI [2]	0.9443
[4]	**0.9472**
[40]	0.9136
[41]	0.8700
[30]	0.9392
[39]	0.8933
[43]	0.8856
[47]	0.8615

Table 4.2: Performance of AdaDot compared to various state-of-the-art methods for PPI on *S. cerevisiae*

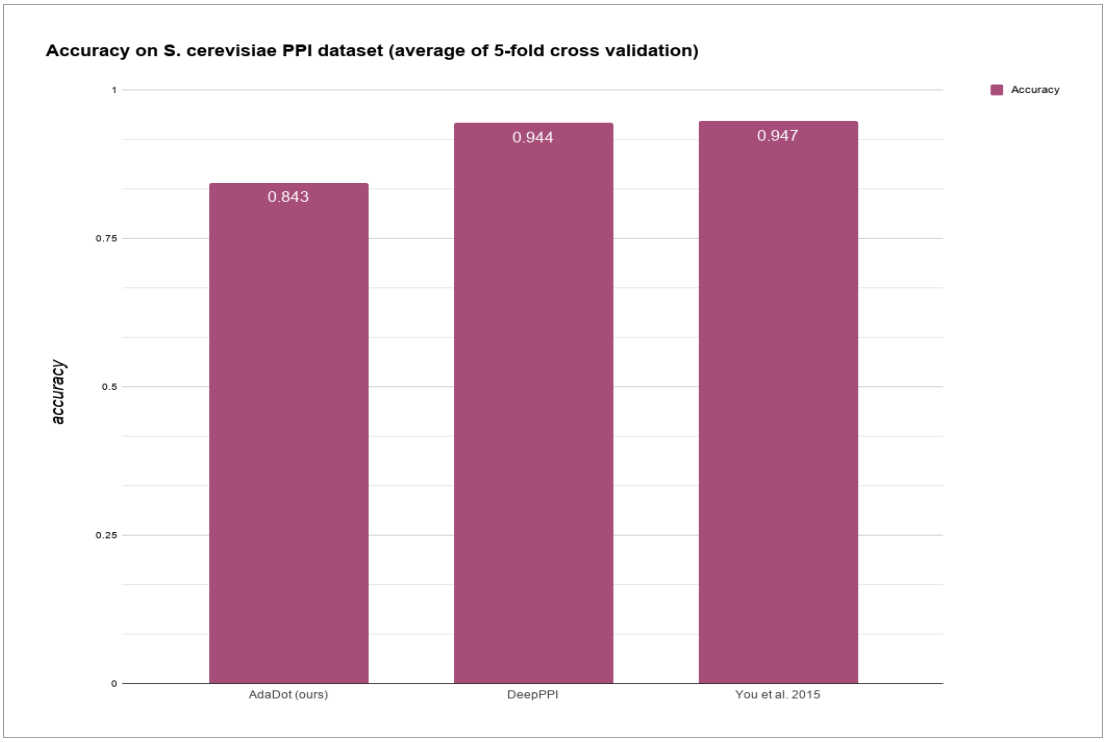

Figure 4.6: Comparison of performance of AdaDot to DeepPPI and the top non-deep method on *S. cerevisiae*

The results of Table 4.1 and Figure 4.5 show that AdaDot is approximately 6% away from the best on *H. pylori*, and the results of Table 4.2 and Figure 4.6 show that AdaDot is approximately 10% away from the best on *S. cerevisiae*. We note that these results are achieved without having to encode any features by hand. We have merely reencoded the amino acid sequence.

During training, it was observed that the validation loss quickly stabilized within just a few epochs. It seems that either there is not enough data for AdaDot to gain much more accuracy. Moreover, we observed that using all of the 24 feature groups did not help much more than using 8 of the most distinguishing features. We therefore anticipate that either using more data or better encodings of the amino acid sequences could further increase the accuracy of our method.

CHAPTER 5
CONCLUSION

5.1 Summary

Current state-of-the-art methods for predicting PPI rely on hand-engineered features. We have shown that convolutions can be used to extract relevant features from an amino acid sequence with minimal feature engineering. Not only can convolutional neural networks extract useful global features, the proposed architecture (AdaDot) is capable of capturing the correct global information and performing nearly as well as the state-of-the-art with minimal reeconding of the amino acid sequence. We therefore observe that AdaDot shows promise at extracting useful sequence information any problem involving comparison of two inputs.

5.2 Future Work

The current architecture of the network is very basic, and does not take advantage of possible improvements such as skip connections or residual blocks. Improving the convolutions could help to increase accuracy. Moreover, it is possible that the 8 feature groupings used to reencode the amino acids are not the best choice. It is likely that better groupings of amino acids could be used to improve the performance of AdaDot on predicting PPI. Most promisingly, we believe that the general AdaDot architecture shows the potential of CNNs for sequence tasks, especially those in which global sequence information is important and two inputs must be compared.

Appendices

APPENDIX A

SUPPLEMENTARY INFORMATION

Here we reproduce the tables containing various clusterings of amino acids according to their known biochemical properties.

ID	Property	Class 1	Class 2	Class 3
1	Hydrophobicity	Polar	Neutral	Hydrophobicity
		RKEDQN	GASTPHY	CLVIMFW
2	Normalized van der Waals volume	0-2.78	2.95-4.0	4.03-8.08
		GASTPD	NVEQIL	MHKFRYW
3	Polarity	4.9-6.2	8.0-9.2	10.4-13.0
		LIFWCMVY	PATGS	HQRKNED
4	Polarizability	0-1.08	0.128-0.186	0.219-0.409
		GASDT	CPNVEQIL	KMHFRYW
5	Charge	Positive	Neutral	Negative
		KR	ANCQGHIL MFPST-WYV	DE
6	Secondary structure	Helix	Strand	Coil
		EALMQKRH	VIYCWFT	GNPSD
7	Solvent accessibility	Buried	Exposed	Intermediate
		ALFCGIVW	PKQEND	MPSTHY
8	Surface tension	-0.20∼0.16	-0.3∼-0.52	-0.98∼-2.46

Table A.1 continued from previous page

ID	Property	Class 1	Class 2	Class 3
		GQDNAHR	KTSEC	ILMFPWYV
9	Protein-protein interface hotspot propensity -Bogan	High (5-21%)	Medium (1.12-3.64%)	Low (0-0.83%)
		DHIKNPRWY	EQSTGAMF	CLV
10	Protein-protein interface propensity -Ma	High (1.21-2.02)	Medium (0.63-1.12)	Low (0.14-0.29)
		CDFMPQRWY	AGHVLNST	EIK
11	Protein-DNA interface propensity -Schneider	High (4-30%)	Medium (1-3%)	Low (0-1%)
		GKNQRSTY	ADEFHILVW	CMP
12	Protein-DNA interface propensity -Ahmad	High (25-100%)	Medium (5-18%)	Low (0-4%)
		GHKNQRSTY	ADEFIPVW	CLM
13	Protein-RNA interface propensity -Kim	High (0.25-11.0)	Medium (-0.25-0.17)	Low (-0.3 - -0.8)
		HKMRY	FGILNPQSVW	CDEAT
14	Protein-RNA interface propensity -Ellis	High (1.18-2.07)	Medium (0.84-1.16)	Low (0.41-0.8)

Table A.1 continued from previous page

ID	Property	Class 1	Class 2	Class 3
		HGKMRSYW	AFINPQT	CDELV
15	Protein-RNA interface propensity - Phipps	High (0.95-1.8)	Medium (0.5-0.95)	Low (0-0.5)
		HKMQRS	ADEFGLNPVY	CITW
16	Protein-ligand binding site propensity - Khazanov	High ($\geq$ 2.25)	Medium (1.6-2.3)	Low (1.5)
		CFHWY	GILNMSTR	AEDKPQV
17	Protein-ligand valid binding site propensity -Khazanov	High ($\geq$ 1.4)	Medium (0.79-1.21)	Low (0.76)
		CFHWYM	DGILNSTV	AEKPQR
18	Propensity for protein-ligand polar & aromatic non-bonded interactions -Imai	High (477-1197)	Medium (95-423)	Low (<95)
		DEHRY	CFKMNQSTW	AGILPV
19	Molecular Weight	Low (75-105)	Medium (115-155)	High (165-204)
		AGS	CDEHIKLMNQPTV	FRWY
20	cLogP	-0.9	-3.07-2.26	-0.73

Table A.1 continued from previous page

ID	Property	Class 1	Class 2	Class 3
		RKDNEQH	PYSTGACV	WMFLI
21	No of hydrogen bond donor in side chain	>1	1	0
		HKNQR	DESTWY	ACGFILMPV
22	No of hydrogen bond acceptor in side chain	>1	1	0
		DEHNQR	KSTWY	ACGFILMPV
23	Solubility in water	High (9-65 g/100g)	Medium (1.14-7.44 g/100g)	Low (0.048-0.82 g/100g)
		ACGKRT	EFHILMNPQSVW	DY
24	Amino acid flexibility index	Very flexible	Moderately flexible	Less flexible
		EGKNQS	ADHIPRTV	CFLMWY

Table A.1: Groupings of amino acids based on different known features (the 8 feature groups used are bolded)

REFERENCES

[1] D. P. Ryan and J. M. Matthews, "Protein–protein interactions in human disease," *Current opinion in structural biology*, vol. 15, no. 4, pp. 441–446, 2005.

[2] X. Du, S. Sun, C. Hu, Y. Yao, Y. Yan, and Y. Zhang, "Deepppi: Boosting prediction of protein–protein interactions with deep neural networks," *Journal of chemical information and modeling*, vol. 57, no. 6, pp. 1499–1510, 2017.

[3] Y. Ding, J. Tang, and F. Guo, "Predicting protein-protein interactions via multivariate mutual information of protein sequences," *BMC bioinformatics*, vol. 17, no. 1, p. 398, 2016.

[4] Z.-H. You, K. C. Chan, and P. Hu, "Predicting protein-protein interactions from primary protein sequences using a novel multi-scale local feature representation scheme and the random forest," *PLoS One*, vol. 10, no. 5, e0125811, 2015.

[5] H. Huang, S. Alvarez, and D. A. Nusinow, "Data on the identification of protein interactors with the evening complex and pch1 in arabidopsis using tandem affinity purification and mass spectrometry (tap–ms)," *Data in brief*, vol. 8, pp. 56–60, 2016.

[6] M. Uhlén, "Affinity as a tool in life science," *Biotechniques*, vol. 44, no. 5, p. 649, 2008.

[7] M. Foltman and A. Sanchez-Diaz, "Studying protein–protein interactions in budding yeast using co-immunoprecipitation," in *Yeast Cytokinesis*, Springer, 2016, pp. 239–256.

[8] P. Y. Chou and G. D. Fasman, "Prediction of protein conformation," *Biochemistry*, vol. 13, no. 2, pp. 222–245, 1974.

[9] M. R. O'Connell, R. Gamsjaeger, and J. P. Mackay, "The structural analysis of protein–protein interactions by nmr spectroscopy," *Proteomics*, vol. 9, no. 23, pp. 5224–5232, 2009.

[10] J. Mehla, J. H. Caufield, and P. Uetz, "Mapping protein–protein interactions using yeast two-hybrid assays," *Cold Spring Harbor Protocols*, vol. 2015, no. 5, pdb–prot086157, 2015.

[11] I. Xenarios, L. Salwinski, X. J. Duan, P. Higney, S.-M. Kim, and D. Eisenberg, "Dip, the database of interacting proteins: A research tool for studying cellular networks of protein interactions," *Nucleic acids research*, vol. 30, no. 1, pp. 303–305, 2002.

[12] H.-W. Mewes, C Amid, R. Arnold, D. Frishman, U. Güldener, G. Mannhaupt, M. Münsterkötter, P. Pagel, N. Strack, V. Stümpflen, *et al.*, "Mips: Analysis and annotation of proteins from whole genomes," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D41–D44, 2004.

[13] G. D. Bader and C. W. Hogue, "Binda data specification for storing and describing biomolecular interactions, molecular complexes and pathways," *Bioinformatics*, vol. 16, no. 5, pp. 465–477, 2000.

[14] H. Hermjakob, L. Montecchi-Palazzi, C. Lewington, S. Mudali, S. Kerrien, S. Orchard, M. Vingron, B. Roechert, P. Roepstorff, A. Valencia, *et al.*, "Intact: An open source molecular interaction database," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D452–D455, 2004.

[15] A. Chatr-Aryamontri, A. Ceol, L. M. Palazzi, G. Nardelli, M. V. Schneider, L. Castagnoli, and G. Cesareni, "Mint: The molecular interaction database," *Nucleic acids research*, vol. 35, no. suppl_1, pp. D572–D574, 2006.

[16] A. J. Enright, I. Iliopoulos, N. C. Kyrpides, and C. A. Ouzounis, "Protein interaction maps for complete genomes based on gene fusion events," *Nature*, vol. 402, no. 6757, p. 86, 1999.

[17] T. Dandekar, B. Snel, M. Huynen, and P. Bork, "Conservation of gene order: A fingerprint of proteins that physically interact," *Trends in biochemical sciences*, vol. 23, no. 9, pp. 324–328, 1998.

[18] S. Wuchty, "Scale-free behavior in protein domain networks," *Molecular biology and evolution*, vol. 18, no. 9, pp. 1694–1702, 2001.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, 2017, p. 3.

[22] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[24] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[25] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint arXiv:1703.10593*, 2017.

[26] G. E. Hinton, "Learning distributed representations of concepts," in *Proceedings of the eighth annual conference of the cognitive science society*, Amherst, MA, vol. 1, 1986, p. 12.

[27] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.*, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 4790–4798.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in *CVPR*, vol. 2, 2017, p. 6.

[30] L. Wong, Z.-H. You, Z. Ming, J. Li, X. Chen, and Y.-A. Huang, "Detection of interactions between proteins through rotation forest and local phase quantization descriptors," *International journal of molecular sciences*, vol. 17, no. 1, p. 21, 2015.

[31] M. Zhang, Q. Su, Y. Lu, M. Zhao, and B. Niu, "Application of machine learning approaches for protein-protein interactions prediction," *Medicinal Chemistry*, vol. 13, no. 6, pp. 506–514, 2017.

[32] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[33] E. Asgari and M. R. Mofrad, "Continuous distributed representation of biological sequences for deep proteomics and genomics," *PloS one*, vol. 10, no. 11, e0141287, 2015.

[34] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, vol. 2, 2015.

[35] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *ICLR*, 2016.

[36] S. Martin, D. Roe, and J.-L. Faulon, "Predicting protein–protein interactions using signature products," *Bioinformatics*, vol. 21, no. 2, pp. 218–226, 2004.

[37] J.-C. Rain, L. Selig, H. De Reuse, V. Battaglia, C. Reverdy, S. Simon, G. Lenzen, F. Petel, J. Wojcik, V. Schächter, *et al.*, "The protein–protein interaction map of helicobacter pylori," *Nature*, vol. 409, no. 6817, p. 211, 2001.

[38] L. Salwinski, C. S. Miller, A. J. Smith, F. K. Pettit, J. U. Bowie, and D. Eisenberg, "The database of interacting proteins: 2004 update," *Nucleic acids research*, vol. 32, no. suppl_1, pp. D449–D451, 2004.

[39] Y. Guo, L. Yu, Z. Wen, and M. Li, "Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences," *Nucleic acids research*, vol. 36, no. 9, pp. 3025–3030, 2008.

[40] Z.-H. You, L. Zhu, C.-H. Zheng, H.-J. Yu, S.-P. Deng, and Z. Ji, "Prediction of protein-protein interactions from amino acid sequences using a novel multi-scale continuous and discontinuous feature set," in *BMC bioinformatics*, BioMed Central, vol. 15, 2014, S9.

[41] Z.-H. You, Y.-K. Lei, L. Zhu, J. Xia, and B. Wang, "Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis," in *BMC bioinformatics*, BioMed Central, vol. 14, 2013, S10.

[42] Y.-A. Huang, Z.-H. You, X. Gao, L. Wong, and L. Wang, "Using weighted sparse representation model combined with discrete cosine transformation to predict protein-protein interactions from protein sequence," *BioMed research international*, vol. 2015, 2015.

[43] Y. Z. Zhou, Y. Gao, and Y. Y. Zheng, "Prediction of protein-protein interactions using local description of amino acid sequence," in *Advances in Computer Science and Education Applications*, Springer, 2011, pp. 254–262.

[44] J. R. Bock and D. A. Gough, "Whole-proteome interaction mining," *Bioinformatics*, vol. 19, no. 1, pp. 125–134, 2003.

[45] L. Nanni, "Hyperplanes for predicting protein–protein interactions," *Neurocomputing*, vol. 69, no. 1-3, pp. 257–263, 2005.

[46] L. Nanni and A. Lumini, "An ensemble of k-local hyperplanes for predicting protein–protein interactions," *Bioinformatics*, vol. 22, no. 10, pp. 1207–1210, 2006.

[47] L. Yang, J.-F. Xia, and J. Gui, "Prediction of protein-protein interactions from protein sequence using local descriptors," *Protein and Peptide Letters*, vol. 17, no. 9, pp. 1085–1090, 2010.